ROBOT MOTION PLANNING

By Jieshan Lu

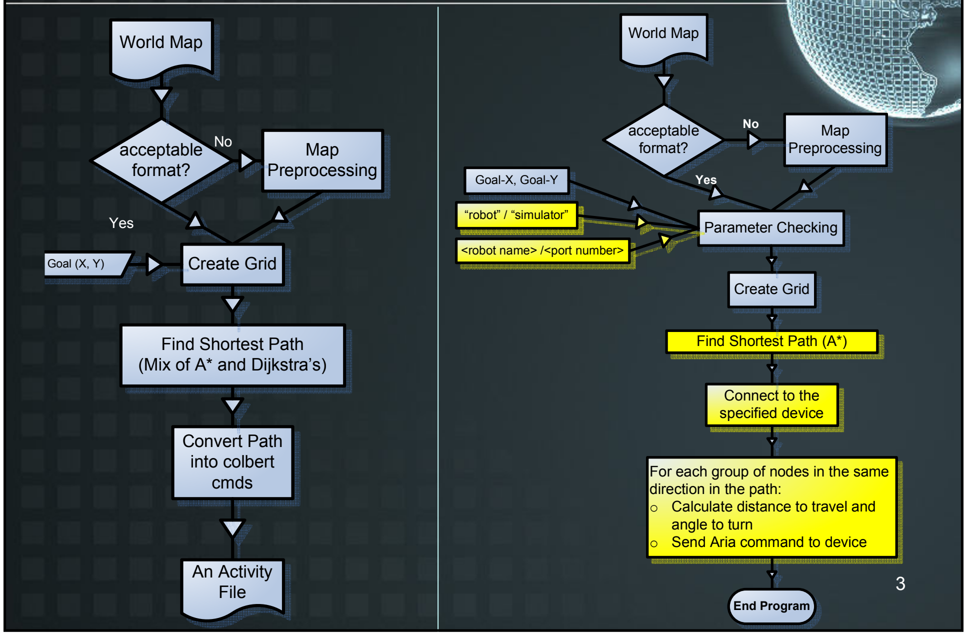
CS 490CA

2005 Winter

OUTLINE

- Old Program vs. New Program
- Back to the A* and Dijkstra's Algorithms
- Research on RoadMap Methods
  - Preliminaries: Graphs, Line Segmentations and Intersections, Convex Hull
  - Visibility Graph
    - General Approach
    - Reduced Visibility Graph
- Demonstration

OLD vs. NEW



3

OLD vs. NEW (cont.)

- Old
  - Path Finding → Human Intervention with the Saphira control interface → Robot motion execution
- New
  - No Human Intervention between path finding and robot motion execution
  - Fully automated

4

OLD vs. NEW (cont.)

- NEW – using “Aria.h”
  - ArTcpConnection tcpConnHdlr;
    ArRobot robot;
  - Aria::init ( )
    - Mandatory initialization – initializes the thread layer and the signal handling methods, also socket layers for windows
  - tcpConnHdlr.open (address, portnum)
    - or tcpConnHdlr.setPort (address, portnum) and openSimple( )
  - robot.setDeviceConnection (&tcpConnHdlr);
    - Used to send commands and receive packets from
  - robot.blockingConnect ( )
    - Return TRUE if connection to robot is established successfully

5

OLD vs. NEW (cont.)

- NEW – using “Aria.h” (cont.)
  - robot.comInt (ArCommands::ENABLE, 1);
  - robot.runAsync (bool stopIfNotConnected);
    - “robot.run (true)” is only good for one-shot programs
    - e.g. actions
  - robot.setDeltaHeading (double degree);
    - robot.isHeadingDone ( )
  - robot.move (double distance /* mm */);
    - robot.isMoveDone ( )
  - robot.stopRunning (true);
  - Aria::shutdown ( );

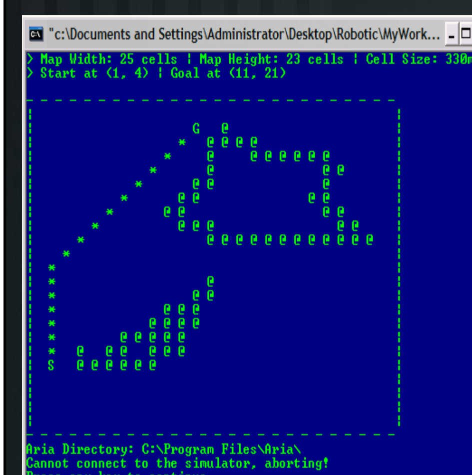
6

A* and DIJKSTRA'S ALGORITHMS

	A* – best first	Dijkstra's – breath first
Given	*S* – a start node *G* – a goal node	*S* – a start node *G* – not necessary
Cost	F(n) = g(n) + h(n)	F(n) = g(n)
G(n)	Cost from *S* to n	Cost from *S* to n
H(n)	• Dx = |Goal_X – n.x|, Dy = |Goal_Y – n.y| •Manhattan – *Dx + Dy* • Euclidean – $\sqrt{Dx^2 + Dy^2}$	Zero
Lists	OpenList – Priority Queue ClosedList	OpenList – Queue ClosedList

7

A* and DIJKSTRA'S ALGORITHMS



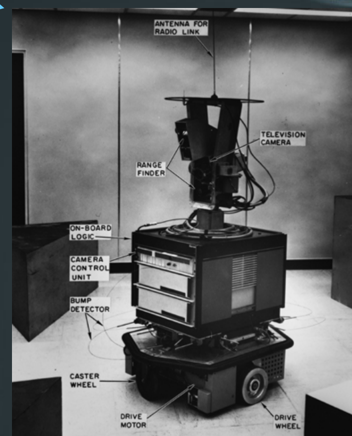
Dijkstra's with Priority Queue

A*

8

Research on RoadMaps

- Introduced in the "Shakey" project at SRI International in the late 60's



- Can produce shortest paths in 2D configuration spaces
- Representations:
  - Visibility Graphs
  - Voronoi Graphs

9

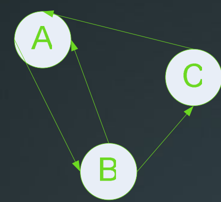
Preliminaries – Graphs

- A graph G = (V, E)
  - V: a set of vertices
  - E: a set of edges
- Directed graph: edges are directional
- Representations:
  - Adjacency Matrix
  - Adjacency List

10

Preliminaries – Graphs

- Adjacency Matrix
  - A [N] [N]
    - where N = |V| number of vertices
  - Space requirement is O ($|V|^2$)
  - Not efficient for large amount of vertices in a given map

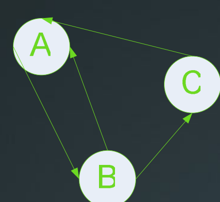


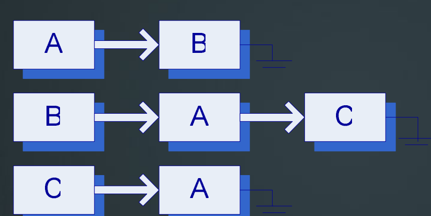
	A	B	C
A	0	1	0
B	1	0	1
C	1	0	0

11

Preliminaries – Graphs

- Adjacency List
  - Standard way
  - Address the space requirement issue for adjacency matrix
  - O (N + M)
    - N = |V|
    - M = # of edges connected to a vertex *v*.

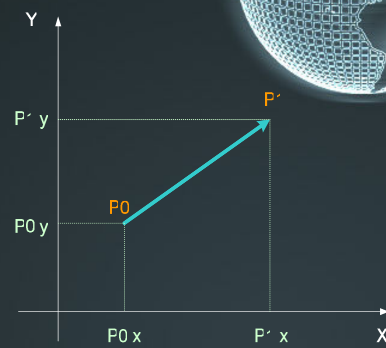




12

Preliminaries – Lines

- A line *L* is defined by two distinct points: P_0 and P_1
- A finite line Segment: $\overline{P_0P_1}$, represents an edge in a graph
- In a 2D environment, *L* takes this representation:



$$L = \frac{X - P_0.x}{P_1.x - P_0.x} = \frac{Y - P_0.y}{P_1.y - P_0.y}$$

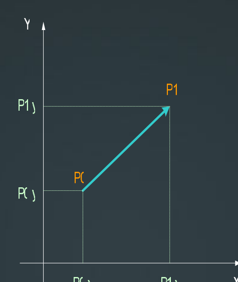
$$\Rightarrow \boxed{(P_1.y - P_0.y)}X + \boxed{(P_0.x - P_1.x)}Y + \boxed{(P_1.x - P_0.x)P_0.y - (P_1.y - P_0.y)P_0.x} = 0$$

$\Rightarrow$ a X + b Y + c = 0

13

Preliminaries – Lines

- For computing intersections of lines and segments in 2D and 3D, it is best to use the parametric equation representation
  - $P(s) = P_0 + s\,(P_1 - P_0) = P_0 + s\,\mathbf{u}$
- where *s* is a real number and **u** = P_1-P_0 is a line direction vector.
- $P(0)=P_0$, $P(1)=P_1$
- when 0<*s*<1, P(*s*) is a point on $\overline{P_0P_1}$, where *s* is the fraction of P(*s*)'s distance along the segment. That is, *s* = d(P0,P(*s*)) / d(P0,P1).
- If *s* < 0 then P(*s*) is outside the segment on the P_0 side
- if *s* > 1 then P(*s*) is outside the segment on the P_1 side.
- For two line intersections, solve s_1 and s_2



14

Preliminaries – Lines

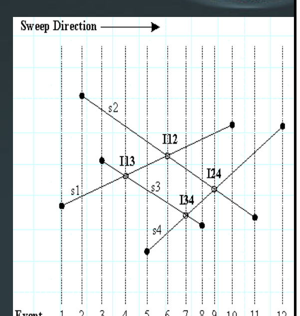
- 2D Line Intersections – very complicated
- For n segments, there are $\binom{2}{n}$ = n(n-1) / 2 = $O(n^2)$ intersection points
- A brute force algorithm is computationally expensive
- Bentley-Ottmann Algorithm – most popular
  - Time: O ((n+k) log n)
  - Space: O (n+k), where k = # of intersections
- There are better algorithms, but they are much more difficult to implement

15

Preliminaries – Lines

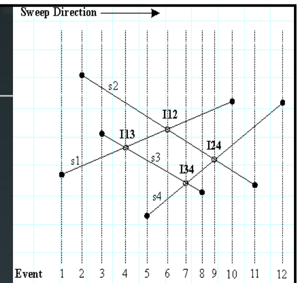
Bentley-Ottmann Algorithm

- Idea: if two segments intersect, so do their x-coordinates
- Input: { L_i }   Output: { I_j }
- Sort input segments by the end points in linear order – increasing x then increasing y
- Event Queue *EQ* maintains a sequence of x values ordered from left to right – endpoints and intersections
- The sweep line *SL* is vertical, sweeps from left to right and stop at certain "events"
  - adds a segment when *SL* encounters the left endpoint – O(log n)
  - deletes it when read in its right endpoint – O(log n)
  - When two segments intersects, their positions are swapped in the list – O(log n) binary search
  - Maintain above-below relationships between two segments
- Two segments intersect only when they are adjacent in SL



16

Preliminaries – Lines



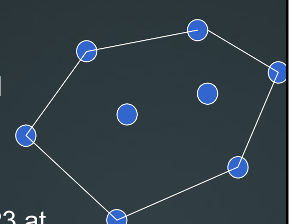
Bentley-Ottmann Algorithm (cont.)

1. EQ (event queue) is empty
2. Insert all segment endpoints into EQ (store segment with left endpoint)
3. Assign SL (status structure) to be empty
4. <u>while</u> (EQ not empty)
   (a) Find next event point p in EQ
   (b) Process event point p
      - remove p from EQ
      - update SL
      - report intersections
      - add new (upcoming) event points To EQ (intersect. pts.)

<u>endwhile</u>

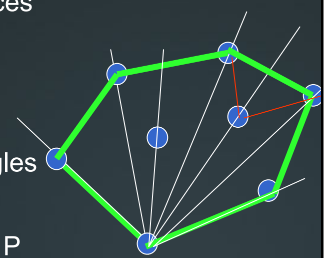
Preliminaries – Convex Hull

- Informal definition: a "rubber band" that is placed around a set of points.
- Properties:
  – All points are either on the hull or inside the hull
  – Each inner angle is less than 180 degree
- Point Orientation Testing:
  – Consider 3 points: P_1 at (x_1, y_1), P2 at (x_2, y_2), P3 at (x_3, y_3)
  – The "Signed Area" function: $\Delta(P_1, P_2, P_3) = x_1y_2 – x_2y_1 + x_3y_1 – x_1y_3 + x_2y_3 – x_3y_2$
  – $\Delta(P_1, P_2, P_3) > 0$ – left turn
    $= 0$ – collinear (a straight line)
    < 0 – right turn



Preliminaries – Convex Hull

- Gift Wrapping Algorithm
  - start with the point with the smallest y, wrap the imaginary rope counterclockwise
  - Average: O(hN); Worst: O (N^2) , h = # of vertices on convex hull, N = # of total points
- Graham Scan Algorithm – most used
  - Start with the point with the smallest y, A
  - Sort remaining points $P_i \in$ {P - {A}} by the angles between (P_i, A) and X-axis – **O(N logN) MergeSort**
  - Scan counterclockwise for the next new point P – O (2N)
    - If P forms a left turn with the last two points in H, or H contains less than two points, add P to H
    - Otherwise, remove the last point in H and re-test p
  - Worst: **O(N logN)**



19

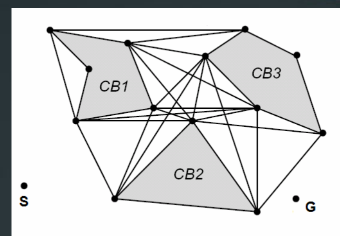
RoadMaps – Visibility Graph

- <u>Input</u>: configurations of *S* and *G*, and a map
- <u>Output</u>: a path in C_{free} connecting *S* and *G*
- <u>Steps</u>:
  1. Build a roadmap in C_{free} (preprocessing – the hard part)
     - roadmap nodes are free or semi-free configurations
     - Two nodes connected by an edge if the robot can move between them
  2. Connect *S* and *G* to roadmap nodes v_s and v_g

20

RoadMaps – Visibility Graphs

- CB – configuration space of an obstacle
- Nodes are connected by an edge if
  - They are an obstacle edge
  - The straight line segment connecting them lies entirely in C_{free}
- Add *S* and *G* as nodes as well



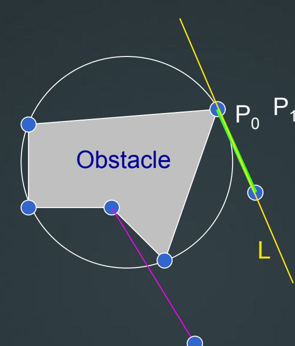
❖ conceptually simple

❖ hard to implement without knowledge about computational geometry

21

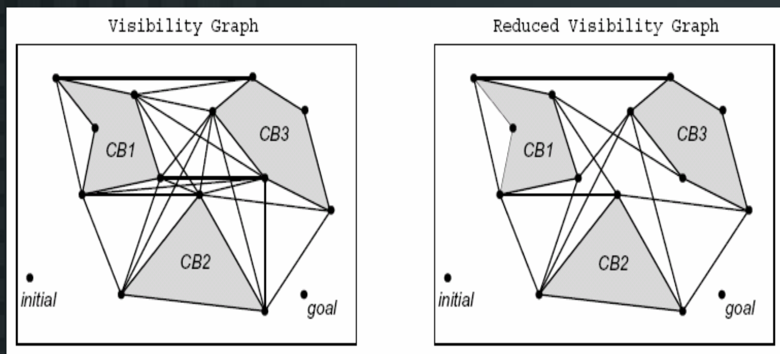
RoadMaps – Reduced Visibility Graphs

- Idea: we don't really need all the edges in the visibility graph
- Easier to build
  1. Construct the convex hull
  2. Construct the pairwise tangents between each obstacles
  - O (N + c^2logN), c = # of obstacles
- All non-obstacle edges are <u>tangent segments</u>
  - Let L be the line passing through an edge (P_0, P_1) in the visibility graph. The segment (P_0, P_1) is a **tangent segment** iff L is a tangent to the obstacle at both P_0 and P_1



22

RoadMaps – Reduced Visibility Graphs



23

Demonstration

24

References

- Wikipedia. “A* search algorithm”. <http://en.wikipedia.org>.
- Amit J.Patel. “Heuristics”. Oct 9, 2004. <http://theory.standford.edu/%7Eamitp/GameProgramming/Heuristics.html>.
- Nancy Amato. “Roadmap Methods”. Randomized Motion Planning. University of Padova. Fall 2004.
- softSufers. “Algorithms”. 2001-2004. <http://softsurfer.com/algorithm_archive.htm>.
- M. T. Goodrich and R. Tamassia. “Convex Hulls”. Algorithm Design. John Wiley & Sons, Inc. New York. 2002. 572 – 582.